Take Your Application Global

Session 141

These are confidential sessions—please refrain from filming or taking pictures.

Application Technologies

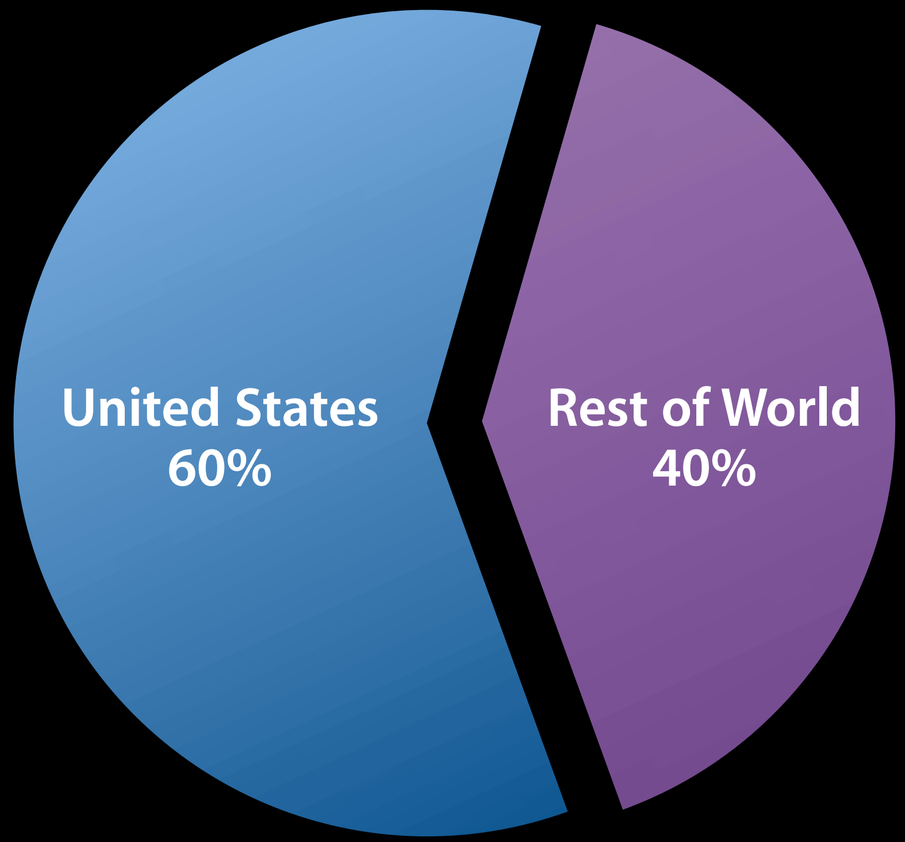
Take Your Application Global

Lee Collins
Manager, OS Engineering Asia

http://developer.apple.com/wwdc2005/

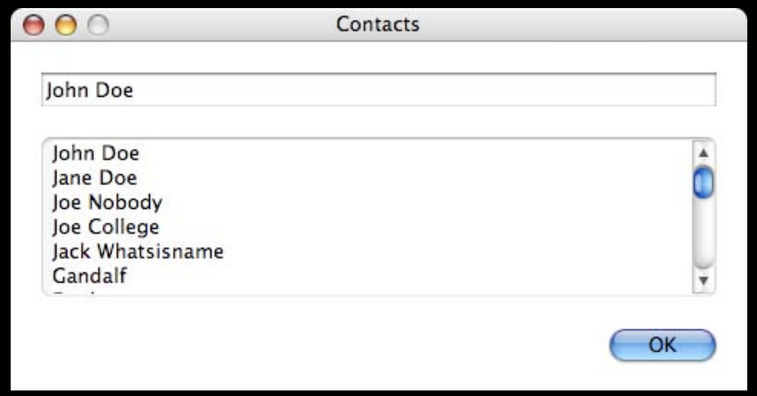
Why Should You Care?

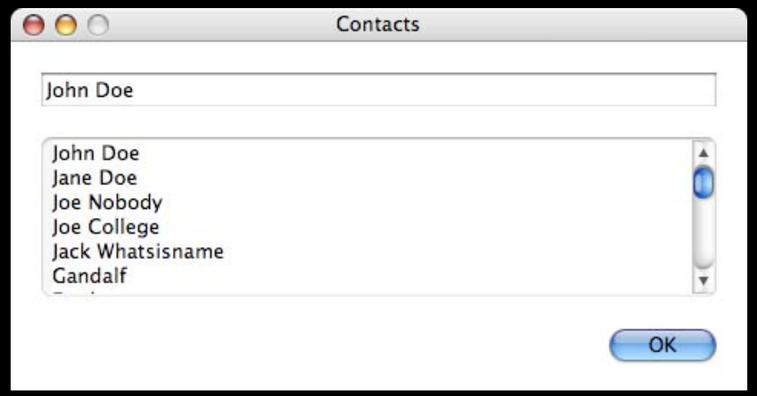
Apple Revenue by Region, Q1 2005

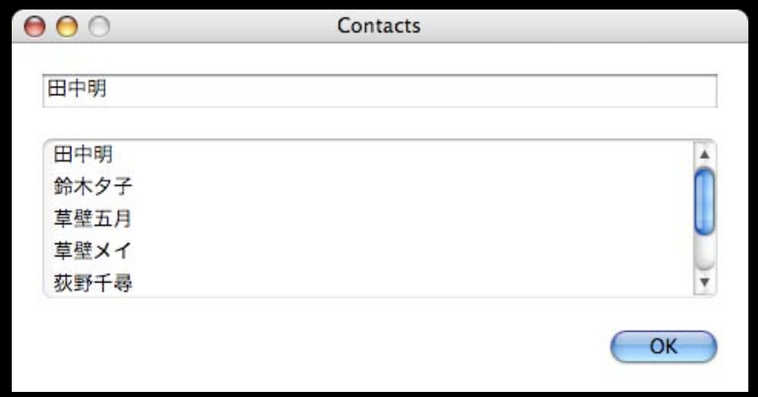


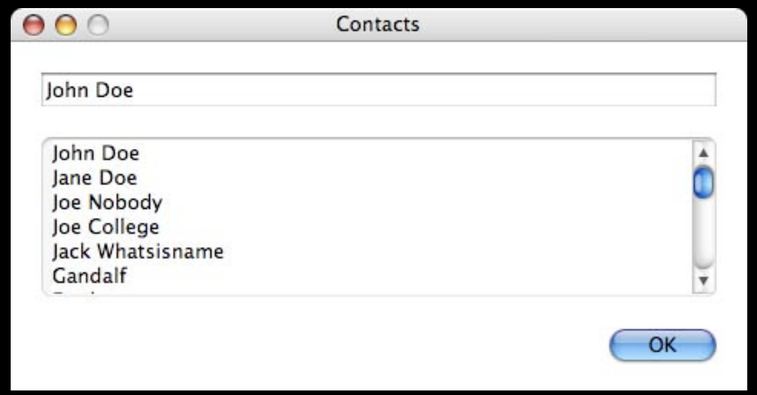
Introduction

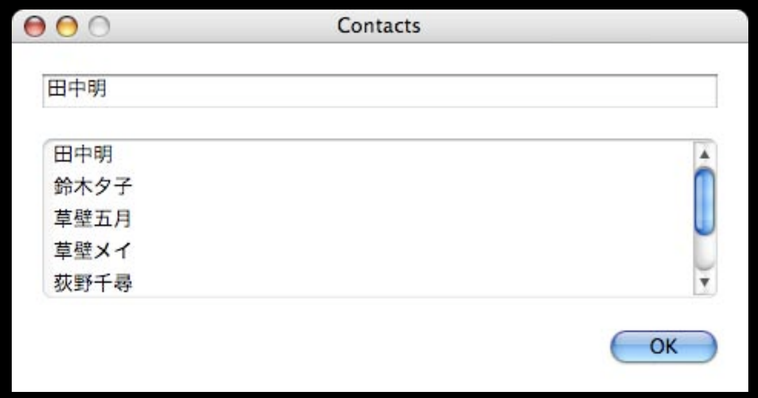
- Localization (“L10N”)
  - Preparing an application for a particular market
- Internationalization (“I18N”)
  - Making an application ready to support any market and any language
- More than translation of user interface
  - Processing of data in multiple languages
  - Support for other countries and cultures
  - Avoiding cultural *faux pas*

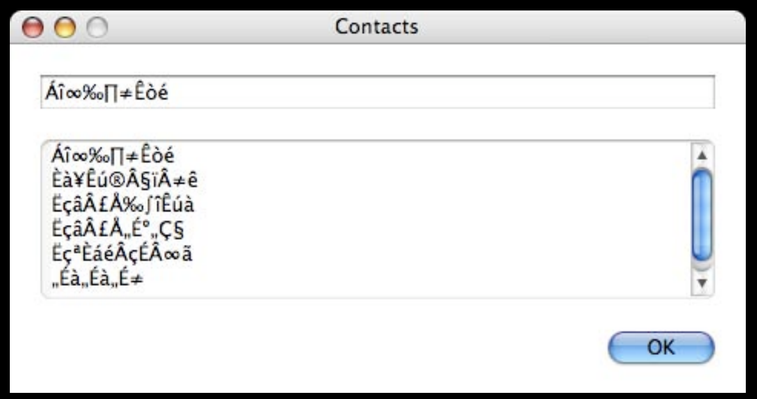

Contacts
John Doe
John Doe
Jane Doe
Joe Nobody
Joe College
Jack Whatsisname
Gandalf
OK


Contacts
John Doe
John Doe
Jane Doe
Joe Nobody
Joe College
Jack Whatsisname
Gandalf
OK


Contacts
田中明
田中明
鈴木夕子
草壁五月
草壁メイ
荻野千尋
OK


Contacts
John Doe
John Doe
Jane Doe
Joe Nobody
Joe College
Jack Whatsisname
Gandalf
OK


Contacts
田中明
田中明
鈴木夕子
草壁五月
草壁メイ
荻野千尋
OK


Contacts
Áî∞‰∏≠Êòé
Áî∞‰∏≠Êòé
È à¥Éú®Â§ïÂ≠ê
ËçâÂ£Å‰∫ïÊúà
ËçâÂ£Å„É°„Ç§
Ëç°ÈáéÂçÉÂ∞ã
„Éà„Éà„É≠

Ideal Global Application

- Handles content in any combination of languages
- Can be localized into any set of languages
- Handles complex and bidirectional writing systems
- Uses local conventions for
  - Dates and times
  - Numbers
  - Calendars
  - Sorting and searching
  - Text boundaries (words, lines, characters)

What You'll Learn

- Which APIs to use for all aspects of your application:
  - Text storage and access
  - Human Interface and localization
  - Drawing, editing, and input
  - Sorting and searching
  - Analyzing and modifying text
  - Dates, times, numbers, and calendars
- Where to look, not detailed recipes
  - "If it's Tuesday, this must be CFString"

International: What's New in Tiger

- OpenType font layout table support
  - If you use Unicode APIs to draw, it just works!
  - Roman and Japanese for Tiger
- String transliteration API

  北京 ⇒ běi jīng
- Unicode type-selection API
- More font Unicode coverage
  - Extended Roman, Extended Greek, Cyrillic in core fonts
  - Tamil, Braille, Yijing Hexagram Symbols, Tai Xuan Jing Symbols
- Tamil input method; Polytonic Greek, Maori, Nepali, other keyboards
- More locale data available (open source— `http://www.unicode.org/cldr/`)

What's New in Tiger, (Cont.)

- New options for number/date/time formatting
  - Number spellout, finer control, calendar calculations
    - "One hundred and twenty-three point four five"
    - `NSCalendar/CFCalendar`
    - `NSDatePicker`
- Non-Gregorian calendars
  - Japanese, Thai Buddhist, Islamic, Hebrew

    January 5, 17 Heisei

    January 5, 2547 BE

    Duh'l-Qi'dah 24, 1425

    Tevet 24, 5765

Demo

John Jenkins

http://developer.apple.com/wwdc2005/

Text Storage and Access

http://developer.apple.com/wwdc2005/

Unicode in 60 Seconds

- One character set for the world
- Over 96,000 characters
- UTF-16: 16 bit units
  - One Unicode code *point* can require one or two UTF-16 code *units*

    丈 U+2000B = 0xD840 0xDC0B
- One user character (”grapheme” or “cluster”) may be many Unicode code points

  R e ◌́ s u m e ◌́ ⇒ Résumé

  V i e ◌̣ ◌̂ t ⇒ Việt
- Multiple “spellings”
  - Pre-composed (NFC) vs. decomposed (NFD) vs. mixture
  - Résumé — Re◌́sume◌́ — Résume◌́

Data Types

- UTF-16 code unit (`UniChar, unichar`)
  - Type for element of UTF-16 array
  - Ok unless you need to process individual code points
- Unicode code point (`UnicodeScalarValue, UTF32Char`)
  - Essential when processing individual code points, e.g. for character properties
- Core Foundation string
  - `CFStringRef, CFMutableStringRef, CFAttributedStringRef`
- Cocoa string (toll free bridge with Core Foundation)
  - `NSString, NSMutableString, NSAttributedString`
- All are native-endian
- External representation is big-endian or has BOM

Storing Your Text

- `CFString, CFMutableString, CFAttributedString` *New!*
- `UniChar *`

`CFStringCreateWithCharacters (NULL, myUniChars, count);`	Create a string from UTF-16 (`UniChar`) array
`CFStringGetCharacterFromInlineBuffer (&buf, index);`	Fast access to string contents
`CFAttributedStringCreate(NULL, cfstring, attributes);`	Create an attributed string with dictionary of arbitrary attributes
`CFAttributedStringGetAttributes (attrString, at, &range);`	Return dictionary of arbitrary attributes at index, and range of run

Finding a Safe Place to Break

- Don’t break in the middle of a user character!

  V|i|e◌̣◌̂|t (Việt)

- For processing/scanning only, not word or line break
- `CFStringGetRangeOfComposedCharactersAtIndex`
- `UCFindTextBreak (kUCTextBreakClusterMask)`

```
// Example: move potential break point to end of cluster
result = CFStringGetRangeOfComposedCharactersAtIndex(myString, index);
CFIndex break = result.location + result.length;
```

Handling External Data

- Older, non-Unicode document formats and Internet Protocols
- Make sure to choose the right encoding:
  - ASCII or UTF-8 for hardwired text (e.g., file names in bundle)
  - CFStringGetSystemEncoding() for legacy plain text documents
  - Based on font script (for each run) for legacy styled text documents
  - Based on MIME (IANA) charset for data from Internet

`CFStringCreateWithCString(NULL, cstr, encoding)`	To Unicode; CFStringCreateWithBytes for more control
`CFStringGetSystemEncoding()`	User's preferred text encoding
`ATSFontFamilyGetEncoding(family)`	Get the text encoding corresponding to a font
`CFStringConvertIANACharSetNameTo Encoding(string)`	Get the text encoding for a MIME charset

Human Interface and Localization

http://developer.apple.com/wwdc2005/

Structuring Your Application

- Application bundle
  - Localized and non-localized files
  - Localized files in *lang*.lproj directory
    - en.lproj, ja.lproj, tlh.lproj
  - Any kind of file: .nib, .strings, .jpeg, .mov, .rtf
- Interface Builder nib files
  - Contain UI elements
  - `CreateNibReference`, `CreateMenuBarFromNib`, `CreateMenuFromNib`, `CreateWindowFromNib`

Getting Localized Strings

- `CFCopyLocalizedString`
- `Localized.strings` file, in UTF-8 or UTF-16

```
// en.lproj/Localized.strings file
"string1" = "Are you sure you want your computer to melt into a puddle?";
"string2" = "Yes";

// ja.lproj/Localized.strings file
"string1" = "コンピューターがメルトダウンしてもいいですか？";
"string2" = "はい";

// Getting the string
CFStringRef warning;
warning = CFCopyLocalizedString(CFSTR("string1"), "CPU meltdown warning");
```

Take Your Application Global

Deborah Goldsmith
International, Unicode Liaison

http://developer.apple.com/wwdc2005/

Drawing and Editing Text

http://developer.apple.com/wwdc2005/

Text Editing Engines

- Multilingual Text Engine—MLTE
  - See “Handling Unicode Text Editing With MLTE”
  - New: can specify encoding when importing, exporting plain text
- `HITextView`
- `EditUnicodeTextControl`

`HITextViewCreate`	Create an HIView that wraps an MLTE object
`HITextViewGetTXNObject`	Get the underlying MLTE object to do other things: save, open, advanced features
`CreateEditUnicodeTextControl`	Creates a control that fully supports Unicode editing

Unicode Text Rendering Is Complex

- Normalization (pre-composed, decomposed, mixed)
- Complex scripts (Thai, Arabic, Indic, others)

  ह ि न् द ी ⇒ हिन्दी

- Bidirectional (Arabic, Hebrew)
- Ligatures, typographic features

  Zapfino ⇒

- Do not attempt at home

Handling Bidirectional Text

- Unicode uses bidirectional algorithm
  - Operates on a paragraph at a time
  - LTR vs. RTL runs based on content
  - Overall direction implicit or explicit
- Unicode APIs like ATSUI or `HIThemeDrawTextBox` takes care of all this for you
- But, must process/draw one paragraph (not run) at a time

Drawing Short Strings

- Labels, etc.

`HIThemeDrawTextBox(cfstring, ...)`	When theme fonts are OK
`TXNDrawCFStringTextBox(cfstring, ...)`	Gives more control over fonts, CGContext, rotation, etc.
`TXNDrawUnicodeTextBox(buffer, count, ...)`	Works with `UniChar *` strings

Drawing Large Amounts of Text

- ATSUI: Apple Type Services for Unicode Imaging
- See “Rendering Unicode Text With ATSUI”
- Opaque objects for:
  - Layout (typically a paragraph or document): `ATSUTextLayout`
  - Styles: `ATSUStyle`
- Easily customizable
- Can get glyph array to use with CoreGraphics

Receiving User Text Input Directly

- `NewTSMDocument` with `kUnicodeDocumentInterfaceType`
- Input via `kEventClassTextInput` CarbonEvents—avoid `kEventClassKeyboard`

`kEventTextInputUnicodeForKeyEvent`	CarbonEvent with Unicode text
`kEventTextInputUpdateActiveInputArea`, `kEventTextInputOffsetToPos`, `kEventTextInputPosToOffset`, `kEventTextInputGetSelectedText`	CarbonEvents for inline input support
`kEventClassTSMDocumentAccess`	CarbonEvents for input method access to document content — better accuracy, more features

Analyzing and Modifying Text

http://developer.apple.com/wwdc2005/

Sorting and Searching

`CFStringCompare(string1, string2, options)`	Compare two CFStrings; use `UCCompareTextDefault` for `UniChar *`
`UCGetCollationKey(collator, text, len, ...)`	Get a key that can be binary compared (best for sorting large data sets)
`CFStringFind(target, search, options)`	Search a target CFString for a search string; also `CFStringFindAndReplace`
`CFStringFindCharacterFromSet(string, set, options, ...)`	Search a string for characters that are members of a `CFCharacterSet`

Transforming Text

- `CFStringTransform(string, transform, &range, reverse)` *New!*

`kCFStringTransformStripCombiningMarks`	Résumé ⟹ Resume
`kCFStringTransformToLatin`	Это будет हिन्दी ⟹ Éto budet hindī
`kCFStringTransformLatinHiragana`	kon'nichiha ⇔ こんにちは
`kCFStringTransformMandarinLatin`	上海 ⇔ shàng hăi
`kCFStringTransformToXMLHex`	Cool!☺ ⇔ Cool!☺

Truncating Text That Is Too…

`HIThemeDrawTextBox`	Truncate to width while drawing; storage unchanged
ATSUI `kATSULineTruncationTag`	Truncate to width while drawing; storage unchanged
`TruncateThemeText`	Truncate string in storage

Finding Text Boundaries

- `TextBreakLocator:UniChar *`

`UCCreateTextBreakLocator`	Create object for locale-sensitive text boundaries: character, cluster, word, line
`UCFindTextBreak`	Find the next or previous text break of the desired kinds
`CFStringGetRangeOfComposed CharactersAtIndex`	Find cluster break in CFString

Line break: Parlez-|vous |français ?

Word break: Parlez-vous français ?

Finding Properties of Characters

- CFCharacterSet
- UCGetCharProperty
- UCIsSurrogateHighCharacter
- UCIsSurrogateLowCharacter
- UCGetUnicodeScalarValueForSurrogatePair

```
// Example: Test whether a character is a decimal digit
UTF32Char char;
digits = CFCharacterSetGetPredefined(kCFCharacterSetDecimalDigit);
if (CFCharacterSetIsLongCharacterMember(digits, char))
    ...
```

Dates, Times, Numbers, and Calendars

http://developer.apple.com/wwdc2005/

Locales

- Represent conventions used for dates, times, numbers, calendars, currency, and units
- Distinct from UI language, collation order, text break behavior

`CFLocaleCopyCurrent()`	User's preferences from Formats pane
`CFLocaleCreateCanonicalLanguageIdentifierFromString`	Put language identifier in canonical form: "English" ⇒ "en"
`CFLocaleCreateCanonicalLocaleIdentifierFromString`	Put locale identifier in canonical form
`CFLocaleCreateComponentsFromLocaleIdentifier`	Pick out pieces of locale identifier; inverse available
`CFLocaleGetValue`	Get data associated with a locale, e.g. decimal
`CFLocaleCopyDisplayNameForPropertyValue`	Return localized name for value: ja ⇒ "Japanese", JP ⇒ "Japan", USD ⇒ "$"

Formatting/Scanning Dates and Times

- `CFDateFormatter`

`CFDateFormatterCreateStringWith AbsoluteTime(NULL, format, time)`	Format a date/time
`CFDateFormatterGetAbsoluteTime FromString(format, string, ...)`	Convert a date/time string to an absolute time
`CFDateFormatterSetProperty (format, key, value)`	Set properties of a CFDateFormatter

```
CFDateFormatterRef format = CFDateFormatterCreate(NULL, CFLocaleCopyCurrent(),
    kCFDateFormatterLongStyle, kCFDateFormatterLongStyle);
CFDateFormatterSetProperty(format, kCFDateFormatterIsLenient, kCFBooleanTrue);
CFDateFormatterGetAbsoluteTimeFromString(format, string, NULL, &time);
```

June 6, 2005

٦ يونيو, ٢٠٠٥

Computing Calendar Data

- CFAbsoluteTime
- CFCalendar New!

CFCalendarComposeAbsoluteTime	Convert a set of calendar values to an absolute time
CFCalendarDecomposeAbsoluteTime	Convert an absolute time to a set of calendar values
CFCalendarAddComponents	Modify individual calendar values that correspond to a CFAbsoluteTime (e.g., add one month)
CFCalendarGetFirstWeekday	Get first day of week for locale
CFCalendarGetRangeOfUnit	E.g., days in month, months in year at specific time; Feb 2000 ⇒ [1,29]

Formatting/Scanning Numbers

- `CFNumberFormatter`

`CFNumberFormatterCreateStringWith Value(NULL, formatter, type, &num)`	Format a number
`CFNumberFormatterGetValueFromString (formatter, string, ...)`	Extract a number from a string
`CFNumberFormatterSetFormat (formatter, format)`	Set format pattern string

```
CFNumberFormatterRef formatter = CFNumberFormatterCreate(NULL, CFLocaleCopyCurrent(),
    kCFNumberFormatterCurrencyStyle);
double cash = 42.0;
string = CFNumberFormatterCreateStringWithValue(NULL, formatter, kCFNumberDoubleType, &cash);
```

42,00 €

৪৪২.০০

Wrap Up

http://developer.apple.com/wwdc2005/

Whom to Contact

Deborah Goldsmith
International Engineer, Unicode Liaison
goldsmit@apple.com

Lee Collins
Manager, OS Engineering Asia
lcollins@apple.com

Xavier Legros
Worldwide Developer Relations
xavier@apple.com

For More Information

http://developer.apple.com/wwdc2005/

Documentation, Sample Code, and Other Resources

http://developer.apple.com/wwdc2005/

Q&A

http://developer.apple.com/wwdc2005/

WWDC2005

Worldwide Developers Conference